



高性能云服务器就选阿里云
20000随机读写IOPS, 256MB/S吞吐量

[查看详情](#)

EF 上班玩手机不如学英语
订阅《每日英语》每天5分钟，碎片时间自我充电

[立即订阅](#)

首页 > Swift

Swift 2.0初探：值得注意的新特性

2015-06-23 10:08 编辑：lansekuangtu 分类：Swift 来源：Dev Talking

0 23105

Swift 2.0 WWDC guard

招聘信息：iOS高级开发工程师



转眼间，Swift已经一岁多了，这门新鲜、语法时尚、类型安全、执行速度更快的语言已经渐渐的深入广大开发者的心。我同样也是非常喜爱这门新的编程语言。

今年6月，一年一度的WWDC大会如期而至，在大会上Apple发布了Swift 2.0，引入了很多新的特性，以帮助开发者能更快，更简单的构建应用。我在这里也说道说道Swift 2.0中值得大家注意的新特性。

guard语句

guard语句和if语句有点类似，都是根据其关键字之后的表达式的布尔值决定下一步执行什么。但与if语句不同的是，guard语句只会有一个代码块，不像if语句可以if else多个代码块。

那么guard语句的作用到底是什么呢？顾名思义，就是守护。guard语句判断其后的表达式布尔值为false时，才会执行之后代码块里的代码，如果为true，则跳过整个guard语句，我们举例来看看。

我们以今年高考为例，在进入考场时一般都会检查身份证和准考证，我们写这样一个方法：

```

1 func checkup(person: [String: String!]) {
2
3     // 检查身份证，如果身份证没带，则不能进入考场
4     guard let id = person["id"] else {
5         print("没有身份证，不能进入考场!")
6         return
7     }
8
9     // 检查准考证，如果准考证没带，则不能进入考场
10    guard let examNumber = person["examNumber"] else {
11        print("没有准考证，不能进入考场!")
12        return
13    }
14
15    // 身份证和准考证齐全，方可进入考场
16    print("您的身份证号为:\(id)，准考证号为:\(examNumber)。请进入考场!")
17
18 }
```

热门资讯

- 分享你最喜欢的技巧和提示(Xcode, Objective-C)

点击量 30315
- 看了一天简历的感悟~心真累

点击量 13517
- 2016年让开发工作变得更简单的小事

点击量 10889
- 实践干货！猿题库 iOS 客户端架构设计

点击量 9406
- 大龄程序员，未来在何方？

点击量 9214
- iOS 架构模式--解密 MVC, MVP, MVVM

点击量 8582
- 我从55个Swift标准库协议中学到了什么？

点击量 6795
- Interface Builder一些使用技巧

点击量 6164
- iOS - 用 UIBezierPath 实现果冻效果

点击量 6047
- 程序猿的年终总结，各种版本各种残

点击量 5748

综合评论

- mark allenzyq 评论了 关于CAShapeLayer的一些实用案例和技巧...
- 抢钱您直说，傻吊光腚 a277283835 评论了 95%手游无版号出版许可证或将洗牌发行...
- 要钱您直说！ scneuzb 评论了 95%手游无版号出版许可证或将洗牌发行...
- 排序 怎么不好使 哪里出错了 sortuniq 1234yws 评论了 让您的Xcode键字如飞...
- mark hmf190195390 评论了 放肆地使用UIBezierPath和CAShape...
- mark

```

19 |     checkup(["id": "123456"]) // 没有准考证，不能进入考场!
20 |     checkup(["examNumber": "654321"]) // 没有身份证，不能进入考场!
21 |     checkup(["id": "123456", "examNumber": "654321"]) // 您的身份证号为:123456, 准考证号为:654321

```

上述代码中的第一个guard语句用于检查身份证，如果检查到身份证没带，也就是表达式为false时，执行大括号里的代码，并返回。第二个guard语句则检查准考证。

如果两证齐全，则执行最后一个打印语句，上面的两个guard语句大括号内的代码都不会执行，因为他们表达式的布尔值都是true。

这里值得注意的是，id和examNumber可以在guard语句之外使用，也就是说当guard对其表达式进行验证后，id和examNumber可在整个方法的作用域中使用，并且是解包后的。

我们再用if else语句写一个类似的方法：

```

1 | func checkupUseIf(person: [String: String!]) {
2 |
3 |     if let id = person["id"], let examNumber = person["examNumber"] {
4 |         print("您的身份证号为:\(id), 准考证号为:\(examNumber)。请进入考场!")
5 |     } else {
6 |         print("证件不齐全, 不能进入考场!")
7 |     }
8 |
9 |     print("您的身份证号为:\(id), 准考证号为:\(examNumber)") // 报异常
10 |
11 | }
12 | checkupUseIf(["id": "123456"]) // 证件不齐全, 不能进入考场!
13 | checkupUseIf(["examNumber": "654321"]) // 证件不齐全, 不能进入考场!
14 | checkupUseIf(["id": "123456", "examNumber": "654321"]) // 您的身份证号为:123456, 准考证号为:654321

```

我们可以看到用if else实现的方法显然不如guard实现的那么精准。而且id和examNumber的作用域只限在if的第一个大括号内，超出这个作用域编译就会报错。

通过上述两个小例子不难看出，guard语句正如一个称职的守卫，层层把关，严防一切不允许发生的事，并且让代码具有更高的可读性，非常棒。

异常处理

在Swift 1.0时代是没有异常处理和抛出机制的，如果要处理异常，要么使用if else语句或switch语句判断处理，要么使用闭包形式的回调函数处理，再要么就使用NSError处理。以上这些方法都不能像Java中的try catch异常控制语句那样行如流水、从容不迫的处理异常，而且也会降低代码的可读性。当Swift 2.0到来后，一切都不一样了。

在Swift 2.0中Apple提供了使用throws、throw、try、do、catch这五个关键字组成的异常控制处理机制。下面我们来举例看看如何使用，我用使用手机刷朋友圈为例。

首先我们需要定义异常枚举，在Swift 2.0中Apple提供了ErrorType协议需要我们自定义的异常枚举遵循：

```

1 | enum WechatError: ErrorType {
2 |     case NoBattery // 手机没电
3 |     case NoNetwork // 手机没网
4 |     case NoDataStream // 手机没有流量
5 | }

```

我们定义了导致不能刷微信的错误枚举'wechatError'。然后定义一个检查是否可以刷微信的方法checkIsWechatOk()：

```

1 | func checkIsWechatOk(isPhoneHasBattery: Bool, isPhoneHasNetwork: Bool, dataStream: Int)
2 |
3 |     guard isPhoneHasBattery else {
4 |         throw WechatError.NoBattery
5 |     }
6 |
7 |     guard isPhoneHasNetwork else {
8 |         throw WechatError.NoNetwork
9 |     }
10 |
11 |     guard dataStream > 50 else {
12 |         throw WechatError.NoDataStream
13 |     }
14 |
15 | }

```

这里注意，在方法名后有throws关键字，意思为该方法产生的异常向上层抛出。在方法体内使用guard语句对各种状态进行判断，然后使用throw关键字抛出对应的异常。然后我们定义刷微信的方法：

```

1 | func playWechat(isPhoneHasBattery: Bool, isPhoneHasNetwork: Bool, dataStream: Int) {

```

colenceli4 评论了 关于CAShapeLayer的一些实用案例和技巧...

mark

mtl168 评论了 关于CAShapeLayer的一些实用案例和技巧...

感谢分享！

breatve 评论了 『零行代码』解决键盘遮挡问题（iOS）...

谢谢收藏了 获取UUID时电话号码没挡上

qx342904511 评论了 iOS—最全的真机测试教程...

不让用英文 咱用拼音行不

gy1680 评论了 手游送审过程中，除了不能出现英文之外还有哪些奇葩事...

相关帖子

BRE BR&E ProMax 2.0.7047.0 工艺流程模拟软件ProMax

请问有没有人做过国际化，是通过下载语言包的方式的？

相册图片旋转了，怎么办

ios招聘

iOS MDM 应用黑白名单

大家在写api的时候用宏多一些还是用const多一些？

百年一遇的问题求解答

为啥有时候模拟器用电脑的键盘输入就不好使？

如何获取某个cell被点击的次数（听说method swizzling可以解决）有大腿懂吗

微博



CocoaChina

加关注

[good]

极光推送 : 极光品牌全面提升！ <http://t.cn/R5iaHKS>

6月13日 13:15 转发(3) | 评论

6月14日 14:41 转发(3) | 评论
CocoaChina主办，有吃有喝有玩有奖，多一些干货，少一些吹牛的线下沙龙来咯！！ 活动介绍：<http://t.cn/R5JiIBx> 报名链接：<http://t.cn/R5JiIBM> 欢迎大家踊跃报名~~名额有限，先报先得~~





三网合一短信通道



```

1 do {
2     try checkIsWechatOk(isPhoneHasBattery, isPhoneHasNetwork: isPhoneHasNetwork, da
3         print("放心刷, 刷到天昏地暗!")
4     } catch WechatError.NoBattery {
5         print("手机都没电, 刷个鬼啊!")
6     } catch WechatError.NoNetwork {
7         print("没有网络哎, 洗洗玩单机吧!")
8     } catch WechatError.NoDataStream {
9         print("没有流量了, 去蹭Wifi吧!")
10    } catch {
11        print("见鬼了!")
12    }
13
14 }
15
16 playWechat(true, isPhoneHasNetwork: true, dataStream: 60) // 放心刷, 刷到天昏地暗!
17 playWechat(true, isPhoneHasNetwork: false, dataStream: 60) // 没有网络哎, 洗洗玩单机吧!
18 playWechat(false, isPhoneHasNetwork: true, dataStream: 60) // 手机都没电, 刷个鬼啊!
19 playWechat(true, isPhoneHasNetwork: true, dataStream: 30) // 没有流量了, 去蹭Wifi吧!
20

```

上述的代码示例中，首先检查是否可以刷微信的方法前使用try关键字，表示允许该方法抛出异常，然后使用了do catch控制语句捕获抛出的异常，进而做相关的逻辑处理。

这套异常处理机制使Swift更加的全面和安全，并且提高了代码的可读性，非常棒。

协议扩展

在Swift 1.0时代，协议（Protocol）基本上类似一个接口，定义若干属性和方法，供类、结构体、枚举遵循和实现。

在Swift 2.0中，可以对协议进行属性或者方法的扩展，和扩展类与结构体类似。这让我们开启了面向协议编程的篇章。

Swift中，大多数基础对象都遵循了CustomStringConvertible协议，比如Array、Dictionary（Swift 1.0中的Printable协议），该协议定义了description方法，用于print方法打印对象。现在我们对该协议扩展一个方法，让其打印出大写的內容：

```

1 var arr = ["hello", "world"]
2 print(arr.description) // "[hello, world]"
3 extension CustomStringConvertible {
4     var upperDescription: String {
5         return "\u{self.description.uppercaseString}"
6     }
7 }
8 print(arr.upperDescription) // "[HELLO, WORLD]"

```

如果在Swift 1.0时代，要想达到上述示例的效果，那么我们需要分别对Array、Dictionary进行扩展，所以协议的扩展极大的提高了我们的编程效率，也同样使代码更简洁和易读。

打印语句的改变

在Swift1中，有'println()'和'print()'两个在控制台打印语句的方法，前者是换行打印，后者是连行打印。在Swift2中，'println()'已成为过去，取而代之的是他俩的结合体。如果你想做换行打印，现在需要这样写：

```
1 print("我要换行!", appendNewline: true)
```

available检查

作为iOS开发者，谁都希望使用最新版本iOS的API进行开发，省事省力。但常常事与愿违，因为我们经常需要适配老版本的iOS，这就会面临一个问题，一些新特性或一些类无法在老版本的iOS中使用，所以在编码过程中经常会对iOS的版本做以判断，就像这样：

```

1 if NSClassFromString("NSURLQueryItem") != nil {
2     // iOS 8或更高版本
3 } else{
4     // iOS8之前的版本
5 }

```

以上这只是一种方式，在Swift 2.0之前也没有一个标准的模式或机制帮助开发者判断iOS版本，而且容易出现疏漏。

在Swift 2.0到来后，我们有了标准的方式来做这个工作：

```

1 if #available(iOS 8, *) {
2     // iOS 8或更高版本
3     let queryItem = NSURLQueryItem()
4
5 } else {
6     // iOS8之前的版本
7 }

```

```

7
8 }

```

这个特性让我们太幸福。

do-while语句重命名

经典的do-while语句改名了，改为了repeat-while：

```

1 var i = 0
2 repeat {
3     i++
4     print(i)
5 } while i < 10

```

个人感觉更加直观了。

defer关键字

在一些语言中，有try/finally这样的控制语句，比如Java。这种语句可以让我们在finally代码块中执行必须要执行的代码，不管之前怎样的兴风作浪。在Swift 2.0中，Apple提供了defer关键字，让我们可以实现同样的效果。

```

1 func checkSomething() {
2
3     print("CheckPoint 1")
4     doSomething()
5     print("CheckPoint 4")
6
7 }
8 func doSomething() {
9
10    print("CheckPoint 2")
11    defer {
12        print("Clean up here")
13    }
14    print("CheckPoint 3")
15
16 }
17 checkSomething() // CheckPoint 1, CheckPoint 2, CheckPoint 3, Clean up here, CheckPoint

```

上述示例可以看到，在打印出“CheckPoint 2”之后并没有打印出“Clean up here”，而是“CheckPoint 3”，这就是defer的作用，它对进行了print("Clean up here")延迟。我们再来看一个I/O的示例：

```

1 // 伪代码
2 func writeSomething() {
3
4     let file = OpenFile()
5
6     let ioStatus = fetchIOStatus()
7     guard ioStatus != "error" else {
8         return
9     }
10    file.write()
11
12    closeFile(file)
13
14 }

```

上述示例是一个I/O操作的伪代码，如果获取到的ioStatus正常，那么该方法没有问题，如果ioStatus取到的是error，那么会被guard语句抓到执行return操作，这样的话closeFile(file)就永远都不会执行了，一个严重的Bug就这样产生了。下面我们看看如何用defer来解决这个问题：

```

1 // 伪代码
2 func writeSomething() {
3
4     let file = OpenFile()
5     defer {
6         closeFile(file)
7     }
8
9     let ioStatus = fetchIOStatus()
10    guard ioStatus != "error" else {
11        return
12    }
13    file.write()
14
15 }

```

我们将closeFile(file)放在defer代码块里，这样即使ioStatus为error，在执行return前会先执行defer里的代码，这样就保证了不管发生什么，最后都会将文件关闭。

`defer`又一个保证我们代码健壮性的特性，我非常喜欢。

Swift 2.0中的新特性当然不止以上这些，但窥一斑可见全豹，Swift 2.0努力将更快、更安全做到极致，这是开发人员的福音，让我们尽情享受这门美妙的语言吧。



微信扫一扫

订阅每日移动开发及APP推广热点资讯

公众号：CocoaChina

[我要投稿](#)

[收藏文章](#)

分享到：

34

[上一篇：Swift 2.0 异常处理](#)

[下一篇：用函数式的 Swift 实现图片转字符画的功能](#)

相关资讯

[【译】Swift 2.0 下面向协议的MVC架构实践](#)

[读懂Swift 2.0中字符串设计思路的改变](#)

[Using Swift with Cocoa and Objective-C \(Swift 2.0中文
在同个工程中使用 Swift 和 Objective-C\(Swift 2.0更新\)](#)

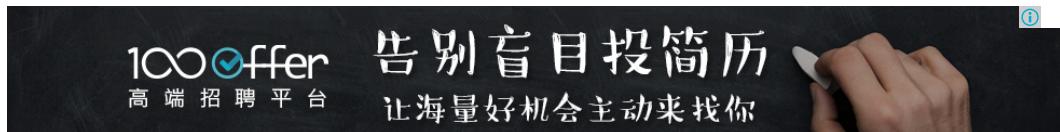
[将 Objective-C 代码迁移到 Swift \(Swift 2.0更新\)](#)

[Swift互用性：采用Cocoa设计模式 \(Swift 2.0版\)](#)

[Swift互用性：与 C 的 API 交互 \(Swift 2.0版\)](#)

[Swift互用性：使用Objective-C特性编写Swift类 \(Swift](#)

[Swift互用性：与 Objective-C 的 API 交互 \(Swift 2.0版更](#)



我来说两句



您还没有登录！请 [登录](#) 或 [注册](#)

所有评论 (0)

[关于我们](#) [广告合作](#) [联系我们](#) [Cocos商店](#)

©2015 Chukong Technologies, Inc.

京ICP备 11006519号 京ICP证 100954号 京公网安备11010502020289



京网文[2012]0426-138号